Revisiting Software Zero-Copy for Web-caching Applications with Twin Memory Allocation

Xiang Song
Jicheng Shi, Haibo Chen and Binyu Zang

IPADS of Shanghai Jiao Tong University
Fudan University
Network I/O Limitations

Network-intensive applications limited by network I/O processing

– Physical limitations
– Efficiency of network sub-systems

*Data copying is one of the key limiting factors*
Data Copy Overhead

Traditional network system calls (e.g., sendmsg) have non-trivial overhead

- Data copying
- Cache thrashing
The Cost of Data Copy

Netperf benchmark using UDP_RR

Execu&on Time (cycles)

Package Size (Bytes)

<table>
<thead>
<tr>
<th>Package Size (Bytes)</th>
<th>Data Copy</th>
<th>Others</th>
<th>Execu&amp;on Time (cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>7000</td>
<td>4000</td>
<td>10000</td>
</tr>
<tr>
<td>64</td>
<td>7000</td>
<td>4000</td>
<td>10000</td>
</tr>
<tr>
<td>128</td>
<td>7000</td>
<td>4000</td>
<td>10000</td>
</tr>
<tr>
<td>256</td>
<td>7000</td>
<td>4000</td>
<td>10000</td>
</tr>
<tr>
<td>512</td>
<td>7000</td>
<td>4000</td>
<td>10000</td>
</tr>
<tr>
<td>768</td>
<td>7000</td>
<td>4000</td>
<td>10000</td>
</tr>
<tr>
<td>1024</td>
<td>7000</td>
<td>4000</td>
<td>10000</td>
</tr>
<tr>
<td>1280</td>
<td>7000</td>
<td>4000</td>
<td>10000</td>
</tr>
</tbody>
</table>

Data Copy: 19.2%
Cache Thrashing Problem

Memcached benchmark

- L2 cache miss Rate
  - 4.58 per thousand cycles (256 Byte)
  - 4.89 per thousand cycles (512 Byte)

<table>
<thead>
<tr>
<th>Top 2 Functions</th>
<th>256 Byte</th>
<th>512 Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>copy_user_generic_string</td>
<td>25.4%</td>
<td>28.7%</td>
</tr>
<tr>
<td>assoc_find</td>
<td>12.8%</td>
<td>10.3%</td>
</tr>
</tbody>
</table>
Challenge in Zero-copy: Data Mutation
Limitations of Existing Solutions

Sendfile and splice
  – Need file back

Fbuf and I/O Lite
  – New API, microkernel oriented

On-demand memory mapping and COW mechanism
  – Protection granularity (e.g., page size)
  – Alignment requirements
Insight into Network Data Mutation

Struct of memcached data

typedef struct _stritem {
    struct _stritem *next;
    struct _stritem *prev;
    ...
    unsigned short refcount;
    ...
    struct { key
        nsuffix
        value}
} item;

Procedure of get request

do_item_get(...) {
    it = assoc_find(key, nkey);
    ...
    if (it != NULL)
        it->refcount++;
    return it;
}

process_get_command() {
    it = do_item_get(key, nkey);
    output_value_data(it);
}
Observation: False Sharing in Protection

Metadata co-locates with the network data

- Modify metadata != modify network data

- False protecting the metadata when protecting the network data
ZCopy

*Idea*: Let applications designate which data should be zero-copied

ZCopy system
- A twin memory allocator
- kernel subsystem

Effective for web caching applications
ZCopy Architecture

Application

Glibc alloc

Data

Data

ZC_alloc

TCP/UDP Stack

ZCopy Proxy

Data Copy

Bypass Data Copy

Hardware
Challenge: Small Memory Blocks

Minimal memory protection
  – Granularity: page size (4 KByte)
  – Alignment: page size

Wasteful to allocate one page for small data blocks
ZCopy Memory Allocator

Aggregating memory blocks with similar sizes

  - Pageblock -- basic memory unit
  - Write protected a pageblock when it is full of zero-copy data
  - Especially friendly to reusable data
    - E.g., cached key/value pairs in memcached
Challenge: Bypass Data Copy

Traditional TCP/UDP network package

```
<table>
<thead>
<tr>
<th>Header</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>addr</td>
<td>length</td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>addr</td>
<td>length</td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>User Space</td>
<td></td>
</tr>
<tr>
<td>Network Data</td>
<td></td>
</tr>
<tr>
<td>Network Data</td>
<td></td>
</tr>
</tbody>
</table>
```
UDP/TCP Package in ZCopy

ZCopy Package

Header

Fragments

User Space

Network Data

Network Data

User Space

Network Data
Package Processing in ZCopy

1. Copy prior none zero-copy data
2. Check user data buffer
3. Handling Zero-copy data (reference)
4. Handling none Zero-copy data (copy)
5. Next user data buffer

Finish
Prototype Implementation

ZCopy is built based on Linux 2.6.38

- A twin memory allocator – ZC_alloc
  - Changed 20 LOCs of streamflow memory allocator
- ZCopy proxy
  - 530 LOCs in UDP and TCP packages processing
- Data protection module
  - 200 LOCs user-level library
  - 205 LOCs kernel module
Experimental Setup

Experimental environment
  – 2 machine with 1.87Ghz Intel Xeon E7 chips
  – Gigabit Network connection
  – Debian GNU/Linux 6.0, Kernel version 2.6.38

Experimental benchmark
  – Memcached
  – Varnish (in paper)
Memcached Setup

Memcached caches multiple key/value pairs in memory

- From a long run’s perspective, the key/value pairs are not expected to be modified or freed
- 10 LOCs of modifications
- Use the memaslap testsuite as client
  - One memcached server using a single CPU core
Memcached UDP Performance

Throughput of Memcached with UDP

Throughput (requests/sec)

Package Size (bytes)

ZCopy
Vanilla Linux

ZCopy Overhead

Network Limitation

128 256 512 768 1024

28.7%
41.1%
Memcached UDP: Package Processing Insight

Package Processing Time

- Vanilla Linux
- ZCopy

Package Processing Time (cycles)

<table>
<thead>
<tr>
<th>Package Size (Bytes)</th>
<th>256</th>
<th>512</th>
<th>768</th>
<th>1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla Linux</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZCopy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Memcached UDP: Cache Misses

L2 Cache Miss Rate

Miss Frequency (1 miss/K cycles)

<table>
<thead>
<tr>
<th>Package Size (Bytes)</th>
<th>UDP Linux</th>
<th>UDP ZCopy</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td></td>
<td></td>
</tr>
<tr>
<td>768</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Memcached TCP Performance

Throughput of Memcahed with TCP

<table>
<thead>
<tr>
<th>Package Size (bytes)</th>
<th>ZCopy</th>
<th>Vanilla Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>40.8%</td>
<td></td>
</tr>
<tr>
<td>768</td>
<td>37.9%</td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td></td>
<td>30.8%</td>
</tr>
<tr>
<td>2048</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Network Limitation

ZCopy Overhead
Future Work

Study and evaluate the performance benefit of ZCopy on other network intensive applications.

Extend ZCopy to efficiently run on multicore machines.

Extend ZCopy to 10 Gigabit network.
Conclusion

This paper presented a new zero-copy system named ZCopy

- A lightweight software zero-copy mechanism based on a twin memory allocator

- Experiments on an Intel machine show that ZCopy outperforms vanilla Linux
Thanks

ZCopy

A lightweight Zero-copy mechanism

Questions?

Institute of Parallel And Distributed Systems
http://ipads.se.sjtu.edu.cn/